Vol.2013-SE-182 No.15
2013/10/24

ドメイン用語辞書の再利用に向けたグループ化

大森 洋一[1]　日下部 茂[1]　林 信宏[1]　荒木 啓二郎[1]

概要：あるドメインにおける用語辞書を関係者が共有することによって，概念の共通理解を得やすくなり，以降のソフトウェア開発工程を効率化することができる．しかし，ドメインにおける用語辞書はいきなり得られるものではなく，当該ドメインにおける複数回の製品開発における要求記述から得られた用語辞書を整理し，蓄積しなければならず，また，蓄積した辞書群を，次の製品開発で再利用しやすい形に整理し，提示されなくてはならない．これまで，このような用語辞書をどのように管理すればよいかという検討はあまりなされていない．本研究では，自然言語とフォーマルメソッドによる意味定義に多対一の関係を許すようにドメインにおける用語辞書を拡張し，用語の分類および各語の統計情報を用いて，再利用しやすいようなグループにまとめるための指針を示し，事例研究により評価する．

キーワード：フォーマルメソッド, 用語辞書, 仕様の再利用

Grouping Technique to Reuse Domain Dictionaries

Abstract: Domain dictionary is a kind of data dictionary which is shared by stakeholders can make them achieve the common understanding of concepts in the domain. Such dictionary may help improving the efficiency of software development in the latter process. The dictionary, however, is not able to be obtained at the first glance. It can be arranbed from the data dictionaries as the results of plural product developments. Therefore, it is important to accumulate and well organize candidates of domain dictionary for reuse, but it has not been studied enough. In this study, we extended the dictionary format to allow many-to-one relationship among terms in a natural language and the definition of them in a formal method,then show guidelines for grouping using statistics of each word and applied it to a case study.

Keywords: Formal method, Domain Dictionary, Specification Reuse

1. はじめに

ソフトウェア開発は，自然言語による要求記述が出発点になる場合が多い．しかし，自然言語による記述には，あいまいさや矛盾が含まれることが多く，それらを解消するには入念なレビューと書き直しが必要であり，なおかつ完全な保証を与えることは難しい．フォーマルメソッドは，計算機システムの仕様を数理的に表記することおよび，その仕様を設計の検証基盤として利用する手法である [6].

フォーマルメソッドを適用し，ソフトウェアの仕様書を数理的なモデルに変換することで仕様書中の矛盾や曖昧さを取り除くことが可能となる．また，フォーマルなモデルはツールにより数学的な検証が可能であり，ソフトウェアの正しさを証明することもできる [14]．このため，特に上流工程からのソフトウェア品質向上に有用であることが知られている．一方，フォーマルメソッドによる仕様記述は，記述言語の習得が必要であったり，背景となる数理的な知識が必要とされたりといった理由により，多くの関係者にとって，不自由なく使いこなしているネイティブな自然言語による記述と比較して，読み手・書き手とも限られる．

したがって，自然言語による仕様記述とフォーマルメソッドによる仕様記述を併用し，それらの間で一貫性が保たれている状態が理想である．実際，高度の信頼性が求められるソフトウェア開発を中心に，さまざまな分野において，フォーマルメソッドの応用が進んでおり，そのほとんどで両者が併用されている [5]．ただし，両者を併用した開発を実施するには 2 種類の仕様書を管理しなければならないという追加作業が発生する．

[1] 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu Univ ersity

ソフトウェア開発においては，開発期間中に仕様の追加や修正といった変更がしばしば発生し，その結果としての，関係者の合意した仕様と自然言語による仕様書と成果物であるプログラムの乖離が問題になっており，既存の手法を拡大適用するだけでは，これらに加えてフォーマルメソッドによる仕様書との一貫性を維持するのは，作業量の面から実現困難である．

我々は，フォーマルメソッドによるソフトウェア開発の普及を目指しており，この理想的な手順の適用対象を広げるために，この 2 種類の仕様書間の一貫性の保証や整合性の検証について，ツールを活用した自動化による作業量低減を検討している．同時に，フォーマルメソッドによる記述はツールを用いた検証が容易になることを利用して，同一ドメインにおけるソフトウェア開発において，自然言語記述に含まれる用語の厳密な意味定義，仕様の再利用による開発効率の改善により，より大きな単位で作業量を相殺できる可能性についても有力であると考えている．

これらを実現するために，ドメイン用語とその定義，および付随する情報を管理するドメイン用語辞書を利用する手法を提案した [15]．従来のデータ辞書は，データベース応用に特化していたり，定義が自然言語で行われているのに対し，我々の研究では用語の意味定義にフォーマルメソッドを用いることで，より一般的な対象に対しても，用語定義の無矛盾性確認や誤解のない意味表現を可能としている．

本稿では，このドメイン用語辞書を管理するツールにおいて，自然言語による記述とフォーマルメソッドによる記述の対応をより多様な場合に適用できるように改善する手法について検討する．特に，仕様の再利用の観点から，実用的な対応付けおよび対応の管理法について検討し，事例研究を行う．以下，第 2 章で再利用を考慮した辞書の構成法，第 3 章ではツールによる実現と拡張について，第 4 章では事例研究およびその成果について，第 5 章ではまとめと今後の課題について述べる．

2. 辞書のグループ化

本研究で対象とする工程は，自然言語とフォーマルメソッドを併用した仕様記述およびその検証である．仕様記述は，曖昧さを含んだ初期の要求記述からはじまり，関係者の合意を取りながら，仕様として確立していく．

よい仕様には

a. 要求が正しく記述されていること

b. 記述の解釈に曖昧さを持たず，簡潔であること

c. 満たすべき要求が全て含まれており，それ以外が含まれないこと

d. 内部に矛盾を含まないこと

e. 要求の優先順位が明確であること

f. 仕様の検証が可能であること

g. 修正が容易であること

h. 仕様を満たしているかどうかの検証が容易であること

といった性質が求められる [4]．

フォーマルメソッドを用いることにより，仕様の満たすべき性質はかなり達成できる．つまり，

A. 数学的な概念は誰でも同じ意味に解釈できる (上記 b)

B. 数学的な矛盾を検出できる (上記 d)

C. 修正の影響範囲が明確である (上記 g)

D. プログラムの意味論を定義可能である (上記 f)

E. 有限状態機械への変換，あるいは変換不能であることの証明が容易 (上記 h)

ただし，これらの言明は仕様記述に用いたフォーマルメソッドの背景となる数理論上の性質に限定される．これに対して，必ずしも数理的な知識があるとは限らない関係者の合意も必要となる上記 a や e，c といった性質の確認については，自然言語による仕様を活用するのが適している．

このような複数の仕様記述の使い分けとそれらの間での一貫性確保を達成するための，我々の基本的なアイデアは以下の手順になる．

(1) 自然言語による仕様をフォーマルメソッドを用いたモデルへ変換し，その上で数理的な検証を行う．

(2) フォーマルなモデルの上で矛盾や記述不足が見つかった場合，それを自然言語記述にフィードバックする．

(3) 修正された自然言語による仕様に同じ手順を繰り返す．

(4) モデル上の問題がなくなったら，自然言語による仕様を関係者がレビューし，妥当性を確認する．

(5) レビューにより修正された自然言語による仕様を修正し，フォーマルメソッドを用いたモデルへ変換し同じ手順を繰り返す．

(6) 確立した自然言語による仕様に含まれる用語とそれに対応するフォーマルメソッドによる定義をドメイン用語辞書として保存する．

(7) 複数の製品開発で得られるドメイン用語辞書の融合，分割，更新など保守を行う．

(8) 同じドメインの問題に対してドメイン用語辞書を再利用する．

このようにひとつの製品の仕様について，2 種類の記述を使い分けるだけでなく，類似の製品からなるドメインについて用語辞書を作成，保守することにより，仕様の再利用によるソフトウェア開発の効率改善を図る．

本研究で想定する仕様記述の手順を図 1 に示す．要求を提示する「顧客」，要求を仕様としてまとめる「仕様記述者」，仕様を詳細化する「設計者」あるいはプログラムを作成する「実装者」といったロールを想定する．「顧客」と「仕様記述者」，「仕様記述者」と「設計者」あるいは「実装者」それぞれの間でのコミュニケーションに適した記述を利用する．それぞれの間にループがあるのは，フォーマルなモデルによる検証とそのフィードバックおよび設計情報の反映による実現可能性のフィードバックを表している．

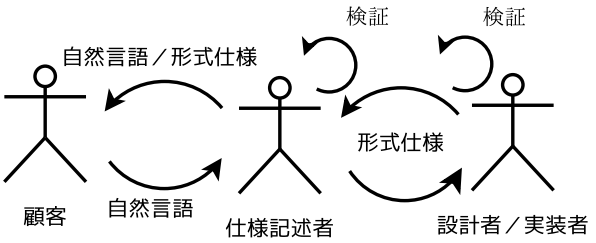


図 1　仕様の洗練過程
Fig. 1　Refinement of Specification

2.1　仕様再利用の課題

効率的な仕様記述には，ドメイン知識の活用が欠かせない．ソフトウェア開発におけるドメインは，複数の製品に共通する性質によって定義される．ドメインの典型的な例は，製品系列である．類似した機能を備えた製品群でありながら，想定する顧客，販売地域，販売時期，製造技術の違いなどにより，複数の製品が開発される場合がある．こうした製品系列では，それぞれの製品に共通する仕様と製品ごとに異なる機能や外観により異なる仕様をもっている．つまり，あるドメインに属する製品の仕様は，ドメイン共通の仕様とそれぞれの製品に固有の仕様からなる．

逆にひとつの製品は，見方により複数のドメインに属するので，一見関係のない製品が共通の性質をもつ場合もある．例えば飛行機の速度や高度をレーダーにより監視しながら管理する管制システムと，センサーによる荷物の確認をしながらベルトコンベアの組み合わせにより配送する集荷システムとは，飛行機あるいは荷物の衝突を避けながら指定の場所へ誘導するという共通の性質をもっており，仕様の再利用が可能である [12]．対象領域の用語の類似性に基づく仕様の再利用といった既存のソフトウェア再利用の技法は，こうした仕様記述の複雑さにより，仕様の再利用にはそのまま適用できない [11]．

本研究では，仕様記述工程における関係者の相互理解の改善および仕様の再利用を想定しており，開発対象の問題を構成する概念を明示的に表現した用語および用語の関連によりドメインを定義する．Web アプリケーションのように計算機リソースの制約が小さいプラットフォームでは，フォーマルメソッドを用いた仕様記述からのツールによるプログラム自動生成も実用的なレベルにあることから，設計や実装の工程はここでは考慮しない．このとき，自然言語による仕様全体とフォーマルメソッドによる仕様全体といった大きな単位での対応付けでは，まったくあるいはほとんど同じ仕様でしか再利用できず，一般的な開発について効率が上がらない．より細かな単位での再利用を考えると，記述が意味を持つ最小単位での管理，すなわち語句の定義と対応付けに行き着く．

この場合の語句は，文法的な意味ではなく，ある仕様において他の語句との相違が識別される最小単位である．すなわち，「りんご」と「それ以外」を選別するシステムでは「りんご」が最小単位であり，「赤いりんご」と「青いりんご」を選別するシステムでは，「りんご」だけでは意味を持たず，「赤いりんご」が最小単位となる．このように，対象となるソフトウェアにより識別されるべき語句は異なり，また文法的な処理だけでは不十分である．

図 1 における仕様記述者は，システムの目的や業務ルールを理解し，顧客からの要求記述に基づいて，よい仕様を記述しなくてはならない．このとき，将来的な拡張可能性など，明確でない部分については自らの直観も活用しなくてはならない．対象システムを特徴づける，自然言語による仕様とフォーマルメソッドによる仕様で対応付けを行うべき語句の選択も，こうした高度に知的な作業の一環であり，作業過程あるいは作業成果に対する制約はできるだけ小さいのが望ましい．

2.2　厳密な用語定義

本研究で使用するドメイン辞書の概念は，構造化分析手法で用いられるデータ辞書の概念に近い．構造化分析はデマルコらによって 1970 年代後半に確立した要求仕様の分析・記述方法であり，データフロー図，データ辞書，プロセス仕様書などを用いてシステムを段階的に詳細化しつつ要求仕様書を作成する [1]．構造化分析におけるデータ辞書は，データフロー図で用いられているデータの構造を記述したもので，用いられるすべてのデータについて記述する必要がある．リアルタイムシステムでの構造化分析で利用するために，データ辞書をフローの定義にまで拡張したものを要求辞書と呼ぶ [3]．こちらの方が本研究で使用する辞書により近い概念である．第 1 章で述べたように，本稿では，フォーマルメソッドを用いて，自然言語記述に含まれる語句の数理的意味論を定義することにより，データ小僧だけでなく関数や手続きに相当する機能やクラスなどの概念も，制限なく表現できる．

同様に類似の概念に用語辞書がある．用語辞書は厳密に

定義されておらず，自然言語処理や知識処理でも使われているが，ソフトウェア開発における仕様に関係した用語としては，開発対象ドメインにおける用語を定義したものを指す．用語辞書は，プログラムの類似性などの分析に用いられたり [8][9]，コンポーネント再利用の手がかりであったり [2][18]，仕様書解釈の補助に用いられたりしている [13]．また，データベース開発の分野ではデータ定義についての用語辞書をデータ辞書と呼び，プログラムの自動生成に利用したりしている [17]．

日本語による識別子を使用したフォーマルメソッドによる仕様書全体を用語辞書とみなす提案もなされている [16]．この提案は，自然言語からフォーマルメソッドへの一方向の変換が想定されており，これに対して，本研究では，語句レベルでの自然言語記述とフォーマルメソッド記述との対応付けを行い，厳密な意味論の定義と自然言語記述も含めた仕様の再利用の両立を目指す．

本研究で使用するドメイン用語辞書は，次のような特徴がある．

- 用語の意味定義にフォーマルメソッドを用いる．数理的な表現により解釈が一意に定まる．
- 辞書に登録する用語 (見出し語) の指定は複数のパターンを許容する．具体的には正規表現によるパターンマッチを行う．
- 見出し語は一連の語を登録することもできる．日本語の自立語は，名詞/代名詞または動詞/形容詞であるのでそれぞれを名詞句，動詞句と呼ぶ．
- 複数の用語の間での一貫性をフォーマルメソッドにより検証可能とする．

さらに，辞書記述の柔軟性を向上させるために，以下の拡張を行った．

2.2.1 階層的な辞書構成

既存のデータ辞書あるいは用語辞書は，人間が読むことが前提となっており，見出し語を辞書順とした一次元の並びのものが多い．フォーマルメソッドを用いても，本質的な依存関係が解決できるわけではない．しかし，フォーマルメソッドによる定義は解釈に曖昧さがないので，ツールによる処理が可能となる．ツールによる処理を前提に，より柔軟性の高い多次元の辞書構造を考えることができる．

具体的には，どのような抽象度に注目するかによってドメインには階層性がある．ドメイン間の静的な関係は，次のように分類できる．

- 継承関係 (is-a) . . . 開発工程の進行による抽象度の違いなどにより，同じ語句の定義がオーバライドされることがある
- 集約関係 (has-a) . . . 複数のサブドメインからなるドメインのドメイン用語辞書は，サブドメインのドメイン用語辞書の集合となる．
- 独立 (independent) . . . ある製品の仕様に互いに依存関係のないドメインも存在する．例えば，かな送りや長音の有無などといった表記のゆれは，問題の性質と無関係に意味の同一性を検証できる．

したがって，ひとつのドメインにひとつのドメイン用語辞書を対応付けて独立して修正・管理することが可能となる．オントロジーの分野では，複数の辞書を用いる場合，用語定義に矛盾を生じる可能性が指摘されている [19]．本研究では，複数辞書間の関連を has-a もしくは is-a に限定し，明示的に保持することで，辞書適用の優先順位を明確に定義する．

2.2.2 文法知識の活用

第 2.1 節で文法知識だけでは，重要な語句の判定は困難であることを述べた．しかし，語句の変化形をそれぞれ登録するのは，同じ意味の語が重複するだけで無駄である．

すなわち，文法知識を利用して，見出し語の活用形についてもパターンマッチさせることができれば，見出し語数を抑制し，登録の手間を削減できる．具体的な例として

- 活用形 . . . 日本語の動詞，形容詞などの用言は活用形をもち，活用の形が決まっているものは，ひとつが登録された時点でさまざまな活用形も登録すれば，辞書作成の効率が向上する．
- 表記のゆれ . . . 日本語では，同じ対象をひらがな，カタカナ，漢字などで表記できる．またスペースや長音の入る場所にもいくつかのパターンがある．このようなゆれは，本来モデル化の前に除かれるべきものであるが，オフショア開発など複数の自然言語での記述に対して意図的に用いることで，自然言語を翻訳する際の意味のずれを避けることができる．ただし，本稿ではこのような場合について，これ以上の検討は行わない．
- 略語 . . . 同じ対象を略語で表記したりする場合，見た目は異なるが同じ物を指すことになるので，辞書の定義は同じエントリに収められるべきである．

のような場合がある．

3. ツールによるサポート

我々は，ドメイン用語辞書を作成する作業をサポートするツールを作成した [15]．本章では，ツールの概要と本稿の検討内容に基づくツールの拡張について述べる．

3.1 ドメイン要求辞書管理ツール

本研究の想定する開発手順では，自然言語と形式モデルの相互変換を頻繁に行なうことになる．このツールは，自然言語による仕様書から形式モデルへの変換を行なう際に，問題記述の検査網羅度を確認するためにキーワードのマーキングを行ない，マークしたキーワードの定義およびキーワードの関連を構造化するドメイン要求辞書の生成を効率的に補助する．

ドメイン要求辞書のひとつのエントリは，見出し語，見

出し語の品詞，自然言語による定義，フォーマルな意味定義，型情報からなる．見出し語の並びは任意であり，後述のモデル生成の場合のみ特定の順序が要求される．

本ツールの利用者は，1 つ以上のファイルからなる要求仕様書を本ツールに読み込ませ，キーワードの網羅度を確認しながらフォーマルなデルへの変換を行ない，マーク付き仕様書と辞書を生成する．本ツールは Eclipse プラグインとして実装しており，さまざまなパースペクティブと組み合わせることができる．図 2 に示すツールの使用画面の例では， Java プロジェクト開発環境と組み合わせて使用している．

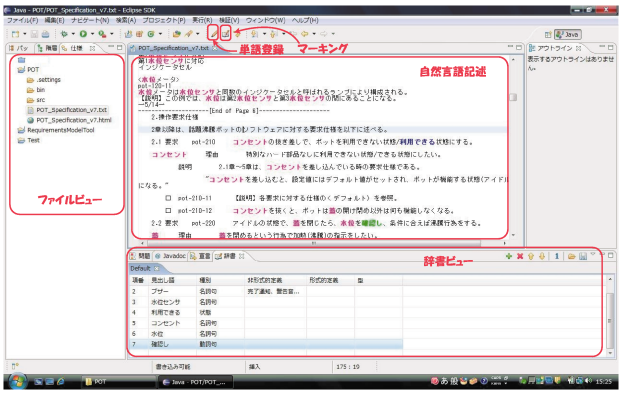


図 2　ドメイン要求辞書管理ツール画面

このツールは

- 自然言語による仕様書に含まれる登録した見出し語をマークする「仕様書マーカー」
- 見出し語にフォーマルな意味定義を与える「辞書編集」
- フォーマルに検証可能なフォーマットを生成する「モデル出力」

の大きく 3 つの機能からなる．

3.2　仕様書マーカー

読み込んだ自然言語記述に含まれるキーワードをマウス等で選択したのち，簡単な登録確定操作により辞書の見出し語として登録する．登録はマウスの右ボタンから呼び出されるメニュー，あるいはツールボタンにより確定し，リアルタイムに指定した辞書に追加される．

「検索」ボタンが押されたら，辞書に登録されている見出し語に対して，自然言語記述中の同一文字列をすべてマークする．設定ファイルにより，名詞句，動詞句，状態といった品詞によって異なる色でマークでき，自然言語記述中のキーワードの視認性を向上させる．マーカーの本質的な機能は，自然言語記述の文字列検索であり，見出し語に Java の正規表現を利用することもできる[*1]．

[*1] Java 正規表現を用いているのは Eclipse の実装による．

3.3　辞書編集

辞書ビューでは，辞書の編集を行う．登録した見出し語には，品詞，自然言語による非形式的定義，形式言語による形式的定義を追加できる．品詞には，名詞句，動詞句，状態の選択肢を用意し，それぞれ，クラスあるいはオブジェクト，関連あるいは操作または関数，状態変数としてフォーマルなモデルへの変換の手がかりとする．

フォーマルな意味定義には VDM++ を用いている．VDM++は形式手法の一種である VDM（Vienna Development Method）の形式仕様記述言語である VDM-SL をオブジェクト指向および非同期通信に関してを拡張した言語である [7]．VDM++ は信頼性を求められるシステムの実開発においても利用され，その成果も報告されている [10]．

本ツールでは，辞書を入力となる自然言語記述と出力となる形式言語記述のマッピングを XML 形式で保存している．したがって，辞書を交換することにより，複数の入力言語あるいは出力言語を切り替えることができる．

このために，辞書の先頭には，

(1)　対象領域
(2)　利用組織
(3)　入力言語
(4)　出力言語

を記述するフィールドがある．対象領域は，ドメインであり，設計対象のみならず，文章検査など任意のドメインを識別する．また，プロジェクトや組織による検査項目の違いからくる見出し語の違いは，利用組織により識別する．入力言語，出力言語は辞書の見出し語の記述言語，形式モデルの記述言語である．

適用する辞書には優先度をつけ，対象ドメインの優先度を表現できる．また同一の優先度で複数の辞書を指定することもできる．

これを利用して，既存のプロジェクトから類似度の高いドメインの辞書を再利用することにより，キーワードの見落としを防ぐことができる．

3.3.1　モデル出力

ドメイン要求辞書に含まれる VDM++ で記述したフォーマルな意味定義をクラス単位でファイルへ出力することにより，　VDM++ の文法や型に関する検証をツールにより行うことができる．

各エントリに書かれたフォーマルな意味定義は，対応する見出し語についてのものであるので，見出し語の並びを変え，ひとつのクラスおよびそのメンバの定義をまとめることで，それらをひとつのファイルへ出力する．

文法や型に関する整合性は，VDMTools により機械的に検証できるので，レビューにおいて自然言語記述のような書き間違いや型の不整合を人間が確認する必要がない [22]．

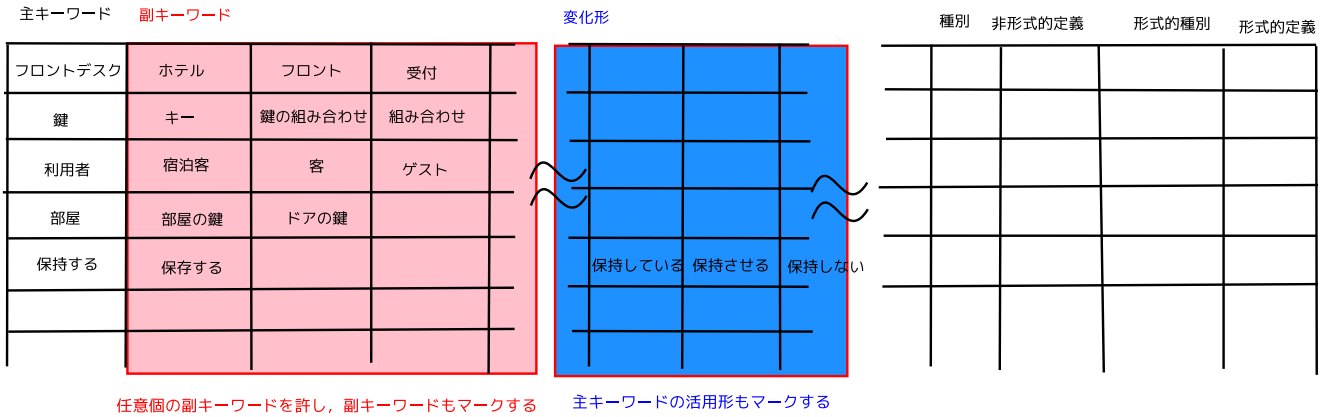


図 3　辞書の拡張

これにより，モデル記述の効率化が可能となる．ここで発見される不整合は，仕様書にフィードバックされ，自然言語の記述を修正する．

3.4　辞書フォーマットの拡張

第 2.2 節における考察に基づいて，ドメイン要求辞書を拡張する．すなわち，現在のツールでは，辞書のエントリとして登録されたキーワードのみが，要求記述においてマークされるので，複数のキーワードがひとつのエントリにまとめられた場合に対応できていない．この問題を解消するために，ひとつのエントリに複数のキーワードを登録可能なように，辞書構造を拡張する．

辞書構造は，ひとつの見出し語を主キーワードとして代表値としてもち，同じエントリで複数のキーワードを登録可能とする．ひとつのエントリは主キーワードに関連した部分と，0 個以上不定個の副キーワードからなる．副キーワードは，主キーワードと同じ意味をもつ語句であり，これもひとつの単語である必要はない．

また，同じエントリーに主キーワードの活用形を保持することも可能とする．

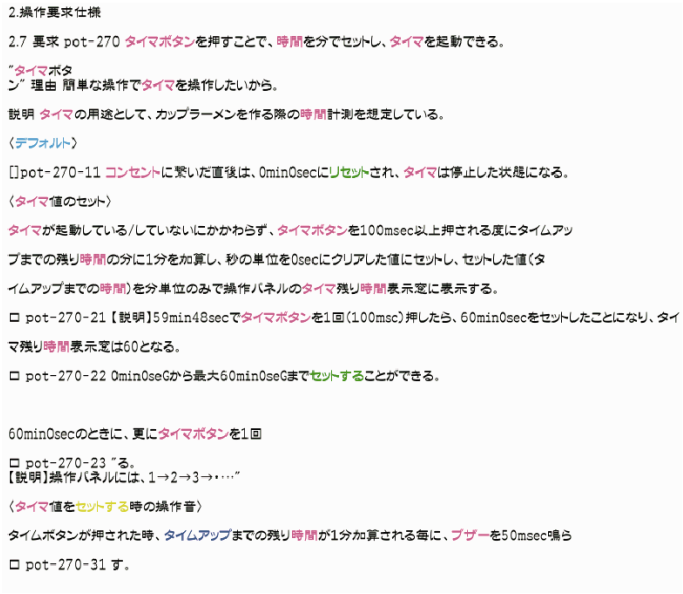
仕様書マーカーは，主キーワード，副キーワード，活用形の全てを同じ品詞としてマークする．また，エントリの削除は代表値のみの削除と，全体の削除を選択可能とする．

図 3 に拡張した辞書のフォーマットを示す．

4.　事例研究

ツールを活用した自然言語記述改善について，SESSAME による「話題沸騰ポット（GOMA-1015 型）要求仕様書 第 7 版」(以下「ポット仕様書」) を例として，組込みソフトウェア開発を例に簡単な評価を行なった [21]．ポット仕様書のキーワードを選択し，マークした結果の一部を図 4 に示す．

我々はこのツールを用いて，仕様の分析，モデル化，VDM++ による検証の後，モデリングの専門家による 3 回のレビューを行ってモデルを洗練し，本ツールが実際の


2.操作要求仕様

2.7 要求 pot-270 タイマボタンを押すことで、時間を分でセットし、タイマを起動できる。

"タイマボタン" 理由 簡単な操作でタイマを操作したいから。

説明 タイマの用途として、カップラーメンを作る際の時間計測を想定している。

〈デフォルト〉

[]pot-270-11 コンセントに繋いだ直後は、0min0secにリセットされ、タイマは停止した状態になる。

〈タイマ値のセット〉

タイマが起動している/していないにかかわらず、タイマボタンを100msec以上押される度にタイムアップまでの残り時間の分に1分を加算し、秒の単位を0secにクリアした値にセットし、セットした値(タイムアップまでの時間)を分単位のみで操作パネルのタイマ残り時間表示窓に表示する。

□ pot-270-21【説明】59min48secでタイマボタンを1回(100msc)押したら、60min0secをセットしたことになり、タイマ残り時間表示窓は60となる。

□ pot-270-22 0min0secから最大60min0secまでセットすることができる。

60min0secのときに、更にタイマボタンを1回

□ pot-270-23 "る。
【説明】操作パネルには、1→2→3→…"

〈タイマ値をセットする時の操作音〉

タイムボタンが押された時、タイムアップまでの残り時間が1分加算される毎に、ブザーを50msec鳴らす。

□ pot-270-31

図 4　電子ポットへの適用

ソフトウェア開発に適用可能であることを確認した [20]．この手順は，通常のモデリングの場合と変わらない．

プログラミング工程については，ポット仕様書はハードウェア要求も含むため，本ツールを利用した VDM++ モデル仕様として Java による話題沸騰ポットのシミュレータを作成している．

本節では，今回の拡張の効果を評価するために，同仕様書に対して形態素解析ツール MeCab [23] を適用し，VDM++ によるモデリング結果と合わせて考察する．

MeCab は 京都大学情報学研究科松本研究室と日本電信電話株式会社コミュニケーション科学基礎研究所の共同研究ユニットプロジェクトを通じて開発されたオープンソース 形態素解析エンジンであり，辞書やコーパスに依存しない形態素解析を高速に行うという特徴がある．

4.1　MeCab 適用結果

ポット仕様書を OCR によりテキスト化したものに MeCab を適用した結果を表 1 に示す．

表 1　Mecab の適用結果

項目	個数	備考
全語数	7581	
全語種	966	同じ語の重複を除いた数
名詞	3855	
名詞の種類	679	同じ語の重複を除いた数
動詞	777	
動詞の種類	157	同じ語の重複を除いた数
形容詞	12	
形容詞の種類	11	同じ語の重複を除いた数

ここでは，自立語を中心に解析した．この結果から次のような考察が得られた．

- 出現頻度と品詞
  出現する語の品詞は，名詞が圧倒的に多い (重複あり: 50.9 %, 重複なし: 70.3%)．以下，動詞 (重複あり: 10.2 %, 重複なし: 16.3%)，形容詞 (重複あり: 0.0%, 重複なし: 1.1%) と続く．
  ただし，出現頻度が高いものには，数値，記号，あるいは「こと」「場合」など一般名詞が含まれており，識別子として用いられる名詞はそれほど多くない．表 2 に名詞の出現頻度の上位を示す．ただし，数値および記号は除いている．
  また，表 3 に動詞，表 4 に形容詞の出現頻度を示す．
- 活用
  動詞については，「する」「した」などの一般的な語が上位を占めているだけでなく，それらの活用形がほとんどである．これは，第 3.4 節で考察した用言の活用形をひとつのエントリにまとめるのが有効であることを示唆する．
  同時に， MeCab による形態素解析が意味を保持するには小さすぎることが分かる．日本語においては「名詞＋する」や「形容詞＋なる」といった動詞句がしばしばみられ，ポット仕様書でも例外ではない．このような語句は分割してはキーワードとしての役割を果たせないので，ひとかたまりの概念として人間が抽出するのが望ましい．
- 表記のゆれ
  ここに挙げただけでも，形容詞のおいて，「なく」と「無く」，「やすく」と「易く」といった表記のゆれが見られる．
  そうした単純な表記の問題だけでなく，名詞には「ポット」が 28 あったが，これと「pot」が同じ概念を指すのかどうかは，自明ではない．

4.2　モデルからの考察

VDM++ によるモデル化に使用した辞書については，表 5 に示す．

このモデルは 7 つのクラスによって構成されており，それぞれが内部状態とメンバ関数をもつ．実際，表 5 には同じ「消灯する」「表示」「状態」といった見出し語が重複しており，それらは実際には異なるクラスに属しているので，異なる概念を表す．ただし，これらは，自然言語による仕様記述では区別さていないので，文脈に依存したやや曖昧な書き方になっている．本来，自然言語による仕様記述へのフィードバックで，「インジケータの表示」「ランプの表示」など異なる概念であることを明示するのが望ましい．

自然言語による仕様記述において，出現頻度の高い語句

表 2　名詞の修験頻度上位

順位	語	個数	順位	語	個数
1	pot	93	11	水位	36
2	温度	63	11	ボタン	36
3	行為	57	13	操作	34
4	沸騰	56	14	タイマ	33
5	センサ	48	15	状態	32
6	要求	44	15	時	32
7	制御	41	17	表示	30
8	説明	40	17	ヒータ	30
9	保温	38	19	給湯	29
10	蓋	37	19	モード	29

表 3　動詞の修験頻度上位

順位	語	個数	順位	語	個数
1	し	98	11	ある	18
2	する	89	12	れ	15
3	た	53	13	い	14
4	ない	27	14	れる	13
5	なっ	23	14	な	13
5	で	23	14	たい	13
7	さ	23	17	できる	12
8	いる	20	18	よう	11
9	なる	19	19	でき	10
9	たら	19	20	ます	9

表 4　形容詞の修験頻度上位

順位	語	個数	順位	語	個数
1	高い	1	2	良い	1
2	なく	1			
2	ない	1			
2	やすく	1			
2	易く	1			
2	近く	1			
2	長く	1			
2	低い	1			
2	熱い	1			
2	無く	1			

表 5　モデル化に利用した辞書

項目	個数	備考
全語数	69	
名詞句	47	
動詞句	17	
状態	5	大域状態変数

は，なんらかの形で，フォーマルなモデルの辞書の見出し語，すなわちモデル化におけるキーワードとして抽出されている．ただし，それは「話題沸騰ポット」のように，名詞句であったり，「満水センサ」，「蓋センサ」，「水位センサ」のように，異なる概念に含まれる語であったりするので，頻度だけでは重要度を判定できない．したがって，文法的な解析による名詞抽出は，モデル化のキーワードの手がかりにはなるが，そのまま使えるわけではない．

動詞についても同様に，ほぼ全てで複合した動詞句がひとつの概念になっている．

5. まとめ

本稿では，ソフトウェア開発における仕様の自然言語記述とフォーマルなモデルの対応づけについて検討した．特に，両方の記述を，意味のある最小単位となる語句レベルで管理する，ドメイン要求辞書として管理する方法について考察した．またこの対応づけの柔軟性を上げるために，開発中のドメイン要求辞書管理ツールの辞書フォーマットを拡張した場合の効果について検討した．

MeCab を利用した形態素解析の結果と，Java によるシミュレータ作成の事例を突き合わせた結果，

- 用言の活用形をまとめるのは辞書のエントリ数を抑えるのに有効であった．
- 名詞の出現頻度を参考にモデル化するには，数値や記号など一般的な語を除く必要がある．
- 類似の表現についても表記のゆれか異なる概念かは自明ではない．

といった考察が得られた．またレビューを重ね，実装可能であったシミュレータのモデルを信頼するならば，そうしたモデルを記述するには，文法的な解析だけでなく人間によるキーワード選択が重要であることも分かった．

他の事例についても同様の検討を行い，仕様の自然言語記述とフォーマルなモデルの関係について，より一般的な考察を行うこと，特に仕様の再利用について実証するのが今後の課題である．

謝辞　事例として，「話題沸騰ポット（GOMA-1015 型）要求仕様書 第 7 版」を提供いただいた SESSAME のみなさま，ツール開発にご協力いただいた吉村康晴氏をはじめ，九州ビジネス株式会社のみなさまに感謝する．本研究の一部は，JSPS 科研費 24220001，基盤研究 (S) 「アーキテクチャ指向形式手法に基づく高品質ソフトウェア開発法の提案と実用化」の成果による．

参考文献

[1] Demarco, T.，高橋 智弘 (訳)，黒田 順一郎 (訳): 構造化分析とシステム仕様，日経 BP (1994).
[2] マヘモド エルホーリー，B.H. ファー，河野善彌: ソフトウェア自動設計におけるコンポーネント再利用のためのデータ辞書，電子情報通信学会技術研究報告 KBSE, Vol. 96, No. 387, pp. 57–64 (1998).
[3] Hatley, D. J. and Pirbhai, I. A.: *Strategies for Real-Time System Specification*, Dorset House (1988).
[4] ieee830: Recommended Practice for Software Requirements Specifications, IEEE Std. 830-1998 (1998).
[5] IPA/SEC: 厳密な仕様記述における形式手法成功事例調査報告書，独立行政法人 情報処理推進機構，http://www.ipa.go.jp/sec/reports/20130125.html (2013).
[6] Jones, C. B.: Software Development based on Formal Methods, *Proceedings of the CRAI Workshop on Software Factories and Ada*, LNCS, Vol. 275, Springer-Verlag, pp. 153–172 (1987).
[7] Jones, C. B.: *Systematic Software Development using VDM*, Prentice-Hall (1990).
[8] 海谷 治彦，北澤 直幸，原 賢一郎，海尻 賢二: 要求変更によるソースコードへのインパクトを分析するシステムの開発と評価，電子情報通信学会論文誌 D， Vol. 93-D, No. 10, pp. 1822–1835 (2010).
[9] 海谷 治彦，北澤 直幸，長田 晃，海尻 賢二: 類似既存システムの情報を利用した要求獲得支援システムの開発と評価，電子情報通信学会論文誌 D， Vol. 93-D, No. 10, pp. 1836–1850 (2010).
[10] 栗田 太郎: 携帯電話組込み用モバイル Felica IC チップ開発における形式仕様記述手法の適用，情報処理， Vol. 49, No. 5, pp. 506–513 (2008).
[11] Maiden, N.: Analogy as a paradigm for specification reuse, *Software Engineering Journal*, Vol. 6, No. 1, pp. 3–15 (1991).
[12] Maiden, N. and Sutcliffe, A.: Exploiting reusable specifications through analogy, *Communications of ACM*, Vol. 35, No. 4, pp. 55–64 (1992).
[13] 中原 俊政，藤野 博之: オフショアプロジェクトにおける UML の適用，プロジェクトマネジメント学会誌，Vol. 10, No. 6, pp. 9–14 (2008).
[14] 中島 震: ソフトウェア工学の道具としての形式手法，NII Technical Report, http://www.nii.ac.jp/TechReports/07-007J.pdf (2010).
[15] 大森 洋一，荒木 啓二郎: 自然言語による仕様記述の形式モデルへの変換を利用した品実向上 に向けて，情報処理学会論文誌：プログラミング， Vol. 3, No. 5, pp. 18–28 (2010).
[16] 佐原 伸: 構造化日本語仕様書としての VDM 仕様，SEC ジャーナル， Vol. 7, No. 1, pp. 23–27 (2011).
[17] 白田 由香里，飯沢 篤志，國井 秀子: データ辞書からのユーザインタフェース生成，情報処理学会研究報告データベースシステム， Vol. 94, No. 62, pp. 89–96 (1994).
[18] 横森 励士，梅森 文彰，西 秀雄，山本 哲男，松下 誠，楠本 真二，井上 克郎: Java ソフトウェア部品検索システム SPARS-J，電子情報通信学会論文誌 D， Vol. 87, No. 12, pp. 1060–1068 (2004).
[19] 横田 一正，緒方 啓孝，劉 渤江: メディエータの探索機能を支援するオントロジーの実現方式，情報処理学会研究報告データベースシステム， Vol. 1998-DBS-116, No. 57, pp. 193–200 (1998).
[20] 井上 心太，大森 洋一，荒木 啓二郎: ツールを使用した形式仕様作成の事例研究，情報処理学会研究報告ソフトウェア工学， Vol. 2012-SE-175, No. 8, pp. 1–8 (2012).
[21] SESSAME WG 2: 組込みソフトウェア開発のためのオブジェクト指向モデリング，翔泳社 (2006).
[22] SCSK システムズ: http://vdmtools.jp/.
[23] 工藤 拓: MeCab: Yet Another Part-of-Speech and Morphological Analyzer, http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html